cLaboratorio

Immagine che contiene testo, schermata, Carattere, nero

Descrizione generata automaticamente



mkdir test //crea cartella

nano main //crea file

rm nomefile

ps ax | grep sshd

ip address (fa vedere tutte le informazioni sulle schede di rete) Per capire se una scheda di rete è attiva, controlliamo che ci sia il tag lower\_up e up.

enp0s3 è il nome della prima scheda di rete di virtual box.

inet dà l'indirizzo IPv4 e inet6 dà l'indirizzo IPv6.

link/ether 08:00:27:7d:fe:48 è il Mac Address.

Clone completo e clone collegato. Il clone collegato crea un’istanza dell’altra macchina senza farla pesare troppo. Scegliamo genera nuovi indirizzi per schede di rete

ip link set dev enp0s8 up serve a abilitare la scheda di rete.

Tramite ssh [user@192.168.56.101](mailto:user@192.168.56.101) accediamo alla macchina virtuale dal cmd

Tramite scp [user@192.168.56.101:pathvirtuale](mailto:user@192.168.56.101:pathvirtuale)nomefile pathhost possiamo spostare le cose dalla macchina virtuale alla macchina ospitante. Il path lo copiamo dalla macchina virtuale usando pwd nel posto della directory.

Se mettiamo \*.c posso prendere tutti i file con estensione .c

Se mettiamo -r dopo scp ma prima del path, trasferiamo un’intera cartella

Conviene spostare il file o farne una copia da un’altra parte.

Per riportalo indietro, riusiamo lo stesso comando al contrario.

I SOCKET

Con la connessione UDP, una volta mandato il pacchetto non sappiamo cosa potrebbe essere successo se è arrivato o meno.   
Con la connessione TCP il problema non si pone perché una volta collegati siamo sicuri di esserci.

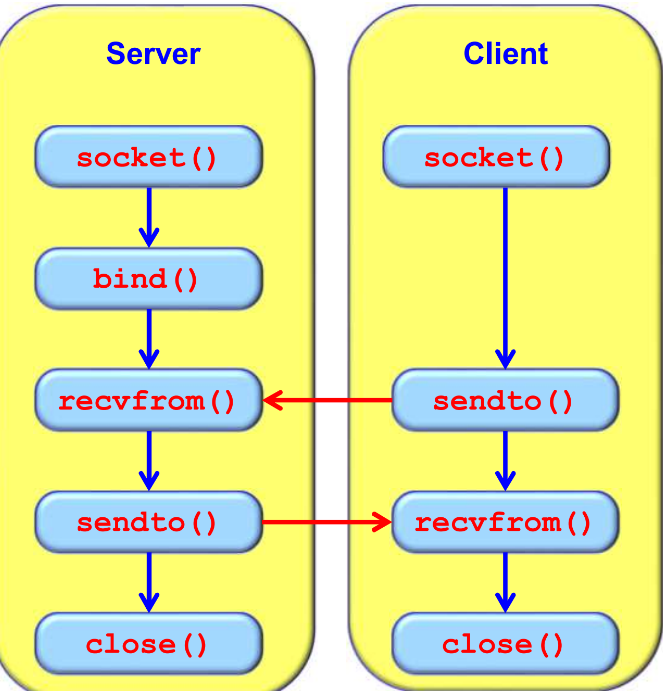
Parlare tra due macchine viene fatto tramite le socket.

I socket sono un’insieme di 5 elementi:

* il protocollo utilizzato
* l’indirizzo ip
* la porta locale
* l’indirizzo ip remoto
* la porta remota

Come protocolli utilizzeremo sempre UDP e TCP, ma le funzioni che vedremo sono state progettate per funzionare per 30 protocolli differenti.

**Connessione UDP**

****

Il client e il server richiamano entrambi una funzione detta socket() che crea una socket che avrà un certo numero. Successivamente solo il server farà una bind() che servirà a chiedere al SO se può aprire una porta x e associarla a lui, se la porta non è possibile associarla a quel processo, restituirà un numero negativo e sarà necessario riaprire il programma. Il client non fa la bind perché sa già il servizio che deve usare. Tuttavia, il client avrà comunque una porta ma sarà già gestita dal SO. Il server fa una recvfrom mentre il client fa sendto. La recvfrom è bloccante, cioè il sistema si blocca ed è in attesa fino a quando non riceve qualcosa. Il client, dunque, manda quindi tramite una sendto un qualcosa. Appena ricevuto il pacchetto, il server fa ciò che deve e manda una risposta al client (non è tenuto a farlo) e il client la riceve tramite una recvfrom. Finito tutto viene effettuata una close per chiudere la socket.

Una prima struct è la sockaddr (pesa 16 byte) la family riporta la tipologia che può essere ipv4 ipv6 o altre. Il data invece conterrà un qualcosa cioè l’ip. Questa struct verrà utilizzata da altre struct per cui noi faremo solamente un cast.

Immagine che contiene testo, schermata, Carattere, Rettangolo

Descrizione generata automaticamente

La struct che useremo per gli IPv4 è AF\_Inet (che peserà 16 byte, dunque essendo che la dimensione è uguale, possiamo tranquillamente effettuare il cast) perché è più comoda da usare. Avremo AF\_INET che conterrà ipv4 o ipv6, sin\_port che conterrà la porta, la struct sin\_addr (che contiene un solo campo addr) e in fine sin\_zero che è un’array di 0. Questo sin\_zero servirà a distinguere bene tutti i campi del pacchetto.

Immagine che contiene testo, schermata, Carattere, linea

Descrizione generata automaticamente

Per le IPv6 utilizzeremo un’altra struct.

Quando mandiamo degli ip, non possiamo mandarli sani come sono, dobbiamo usare delle cose particolari. Convertiamo da Presentation to Network. La funzione che useremo è \*inet\_ntop. Anche in questa funzione dobbiamo specificare il tipo di ip che stiamo mandando.

Funzione da Network to presentation

Immagine che contiene testo, schermata, Carattere, linea

Descrizione generata automaticamente

La funzione Socket è la prima che dobbiamo fare per aprire a tutti gli effetti la comunicazione

Immagine che contiene testo, Carattere, schermata, Blu elettrico

Descrizione generata automaticamente

Immagine che contiene testo, schermata, Carattere

Descrizione generata automaticamente

Il protocollo intende la famiglia e se impostiamo zero, questo verrà settato in automatico in base ai primi due parametri.

Il server dovrà fare la bind.

Immagine che contiene testo, schermata, Carattere, linea

Descrizione generata automaticamente

Restituisce 0 in caso di successo. In addrlen dobbiamo inserire la dimensione della struttura. Già qui dovremo andare a fare un cast da sockaddrin a sockaddr.

Dopo dobbiamo impostare la sendto e la recvfrom. Queste due funzioni sono praticamente uguali.

Immagine che contiene testo, Carattere, schermata, linea

Descrizione generata automaticamente

Finite tutte le operazioni faremo la close.

|  |  |
| --- | --- |
| **Client** | **Server** |
| Socket() | Socket() |
|  | Bind() |
| SendTo(msg) | Recvfrom() {riceverà msg, ip e porta} |
| Close() | Close() |

Il client dovrà prendere sicuramente un Indirizzo Ip e una porta del server, questi li inseriremo nella struct sockaddr\_in e poi in un sbuffer[100].

Al server servirà sicuramente una variabile P dove mettere il numero di porta e un buffer di memoria, e un’altra sock addr in per il client

Possiamo dare IP stabili e IP volatili (all’esame meglio usare quelli stabili)

Le schede di rete impostate a rete interna se hanno lo stesso nome sono nella stessa LAN. Se vogliamo fare comunicare due MV queste devono appartenere alla stessa LAN e devono degli indirizzi IP validi. Potremmo abilitare una scheda di rete solo per passare i file dal PC alla VM.

Per abilitare una scheda di rete dobbiamo fare ip link set dev nome up

Possiamo configurare le schede di rete così tramite il path etc/network/interfaces

Immagine che contiene testo, elettronica, schermata, schermo

Descrizione generata automaticamente

Immagine che contiene testo, schermata, diagramma, linea

Descrizione generata automaticamente

Tramite il diagramma albero possiamo “assegnare gli indirizzi ip”.

Per permettere il routing in IPv4 dobbiamo modificare in etc/sysctl.conf per IPv6 quella sotto

Immagine che contiene testo, schermata, Carattere, design

Descrizione generata automaticamente

